Basic EDA with Healthcare Twitter Analysis datasets

Drawn from Matthew A. Russell *Mining the Social Web*

Find the top ten words, users, hashtags, urls

note: stop words are words, even if unloved

In [9]:

```
#
# The "find_WordsHashUsers" function is on the GitHub repo
#
# It pulls all the individual words, hashtags, user-mentions and URLs
# out of any of the *.csv files provided for this project
#
# It can return either a list of these: every single occurence,
#                       or a set: each entry is unique
#
# Both forms of this function are used in this IPython notebook
#
from find_WordsHashUsers import find_WordsHashUsers

word_list, hash_list, user_list, url_list, num_tweets = \
           find_WordsHashUsers("../files/Tweets_Celiac_full.csv", "content", 
"list")

from collections import Counter

for item in [word_list, user_list, hash_list, url_list]:
    c = Counter(item)
    print c.most_common()[:10] # top 10
    print
```

```
[('rt', 1734), ('to', 1675), ('the', 1536), ('a', 1436), ('for', 1349), ('of'
, 973), ('you', 966), ('is', 958), ('with', 881), ('and', 863)]

[('celiacbeast', 237), ('jenniferswayje', 206), ('thedailyshow', 133), ('glut
endude', 131), ('gfreeradio', 110), ('gfreeschool', 79), ('glutinofoods', 70)
, ('udisglutenfree', 68), ('celiacawareness', 66), ('rudisglutenfree', 56)]

[('celiac', 5505), ('glutenfree', 2933), ('coeliac', 788), ('gf', 705), ('glu
ten', 525), ('gfree', 266), ('health', 194), ('abcdrbchat', 155), ('college',
 128), ('foodallergy', 115)]

[('http://t.co/kliv5zfntq', 79), ('http://t.co/5rty8ts2rh', 67), ('http://t.c
o/qnttxkvxy1', 44), ('http://t.co/bne5lmo8zr', 43), ('http://t.co/ve9kfykgle'
, 42), ('http://t', 40), ('http://t.co/p9a8ezcnm7', 36), ('http://t.co/g2ztri
mtzn', 33), ('http://t.co/vviypdhwxh', 24), ('http://t.co', 21)]
```

Nicer display of the top ten

In [2]:

```
from prettytable import PrettyTable

for label, data in (('Word', word_list),
                    ('Screen Name', user_list),
                    ('Hashtag', hash_list),
                    ('URL', url_list)):
    pt = PrettyTable(field_names=[label, 'Count'])
    c = Counter(data)
    [ pt.add_row(kv) for kv in c.most_common()[:10] ]
    pt.align[label], pt.align['Count'] = 'l', 'r' # Set column alignment
    print pt
```

```
+------+-------+
| Word | Count |
+------+-------+
| rt   |  1734 |
| to   |  1675 |
| the  |  1536 |
| a    |  1436 |
| for  |  1349 |
| of   |   973 |
| you  |   966 |
| is   |   958 |
| with |   881 |
| and  |   863 |
+------+-------+
+-----------------+-------+
| Screen Name     | Count |
+-----------------+-------+
| celiacbeast     |   237 |
| jenniferswayje  |   206 |
| thedailyshow    |   133 |
| glutendude      |   131 |
| gfreeradio      |   110 |
| gfreeschool     |    79 |
| glutinofoods    |    70 |
| udisglutenfree  |    68 |
| celiacawareness |    66 |
| rudisglutenfree |    56 |
+-----------------+-------+
+-------------+-------+
| Hashtag     | Count |
+-------------+-------+
| celiac      |  5505 |
| glutenfree  |  2933 |
| coeliac     |   788 |
| gf          |   705 |
| gluten      |   525 |
| gfree       |   266 |
| health      |   194 |
| abcdrbchat  |   155 |
| college     |   128 |
| foodallergy |   115 |
+-------------+-------+
+-----------------------+-------+
| URL                   | Count |
+-----------------------+-------+
| http://t.co/kliv5zfntq |    79 |
| http://t.co/5rty8ts2rh |    67 |
| http://t.co/qnttxkvxy1 |    44 |
| http://t.co/bne5lmo8zr |    43 |
| http://t.co/ve9kfykgle |    42 |
| http://t                |    40 |
| http://t.co/p9a8ezcnm7 |    36 |
| http://t.co/g2ztrimtzn |    33 |
| http://t.co/vviypdhwxh |    24 |
| http://t.co             |    21 |
+-----------------------+-------+
```

Lexical Diversity and per-Tweet Averages

In [3]:

```
word_set, hash_set, user_set, url_set, num_tweets =  \
           find_WordsHashUsers("../files/Tweets_Celiac_full.csv", "content",
"set")

# A function for computing lexical diversity
def lexical_diversity(set_, list_):
    return 1.0*len(set_)/len(list_)

# A function for computing the average number of entity per tweet
def average_words(list_, num_tweets):
    return 1.0*len(list_)/num_tweets

print "Lexical Diversity"
print "words  %0.2f"%lexical_diversity(word_set, word_list)

print "hashes %0.2f"%lexical_diversity(hash_set, hash_list)
print "users  %0.2f"%lexical_diversity(user_set, user_list)
print "urls   %0.2f"%lexical_diversity(url_set, url_list)

print "\nAverage per tweet"
print "words  %0.2f"%average_words(word_list, num_tweets)
print "hashes %0.2f"%average_words(hash_list, num_tweets)
print "users  %0.2f"%average_words(user_list, num_tweets)
print "urls   %0.2f"%average_words(url_list, num_tweets)
```

```
Lexical Diversity
words  0.13
hashes 0.09
users  0.23
urls   0.64

Average per tweet
words  11.76
hashes 3.11
users  0.88
urls   0.76
```

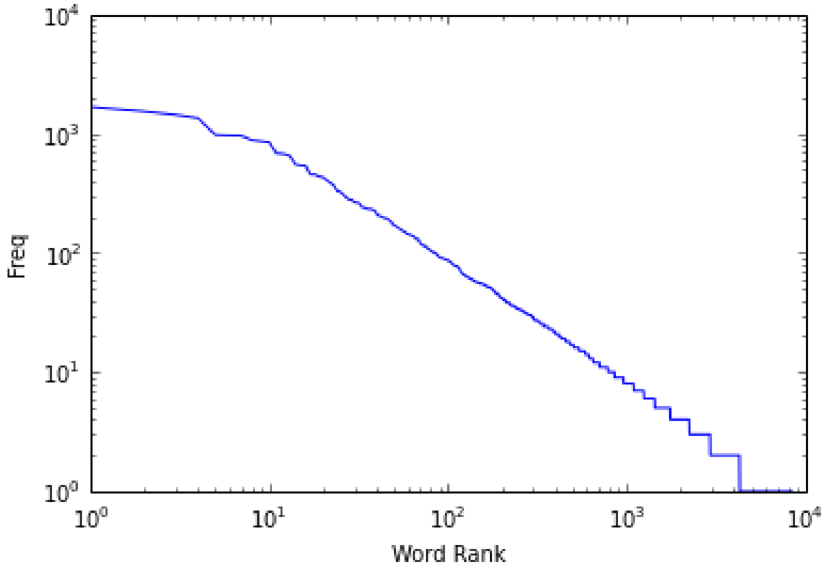
Frequency Binning

In [4]:

```
word_counts = sorted(Counter(word_list).values(), reverse=True)

plt.loglog(word_counts)
plt.ylabel("Freq")
plt.xlabel("Word Rank")
```

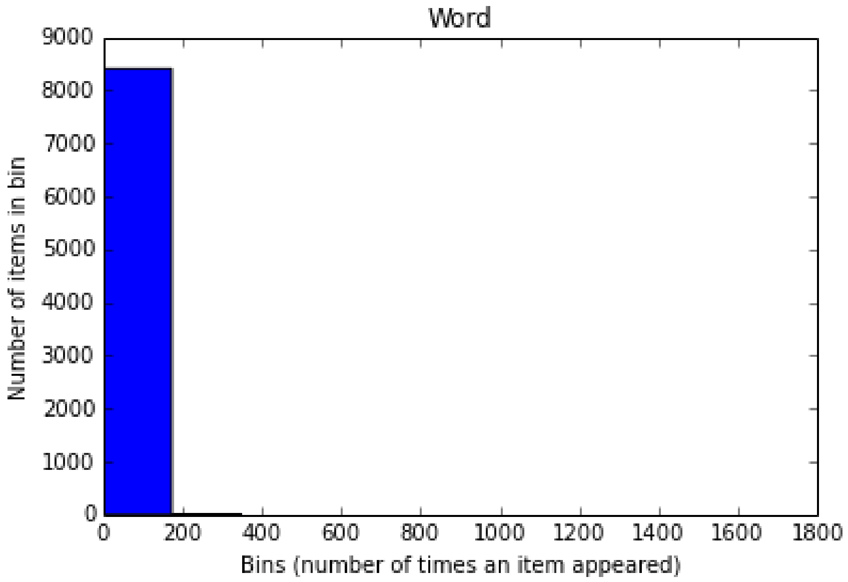
Out[4]:

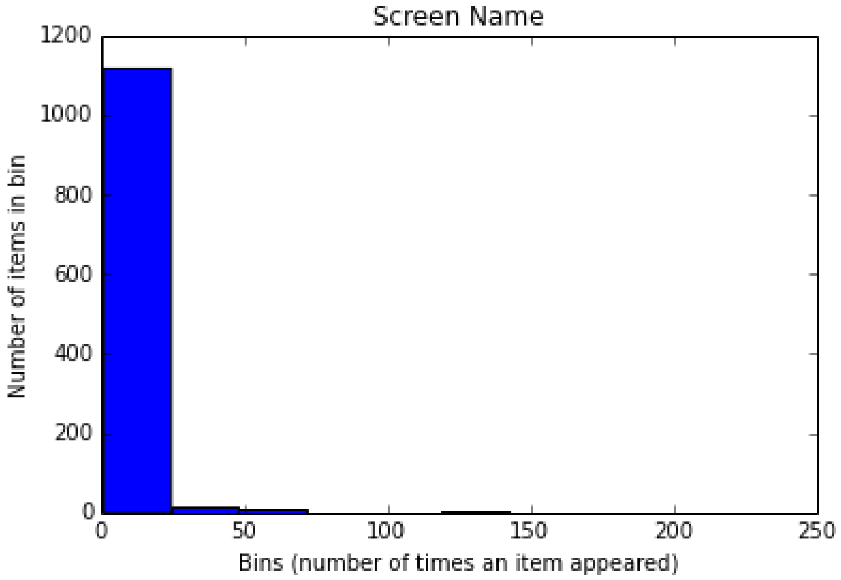
```
<matplotlib.text.Text at 0xe407be0>
```

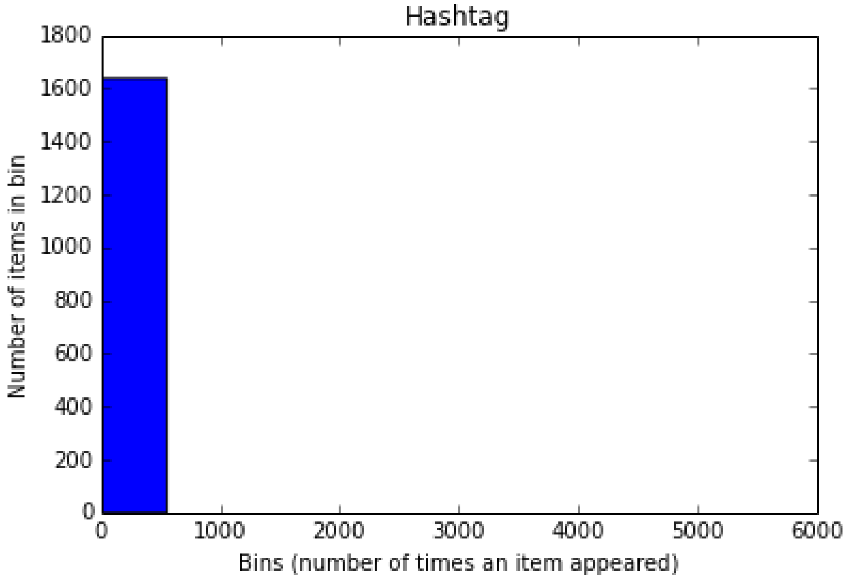


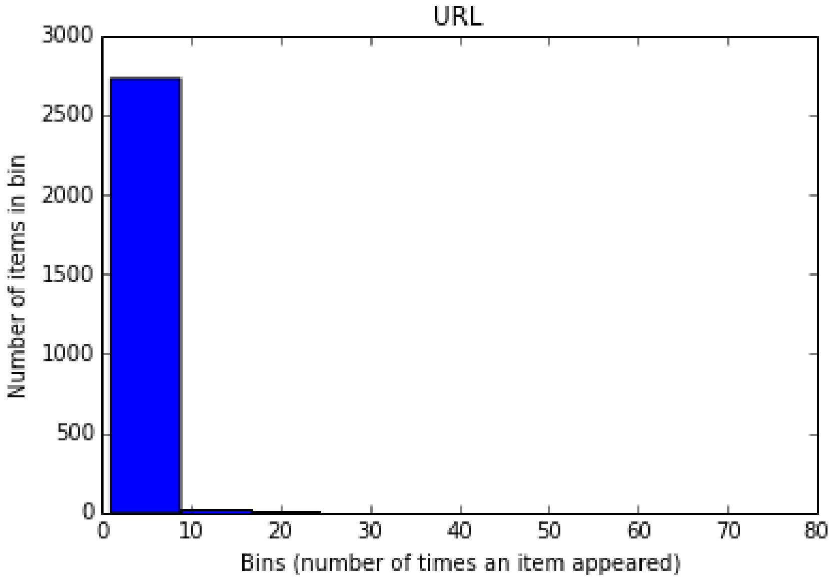
In [5]:

```
for label, data in (('Word', word_list),
                    ('Screen Name', user_list),
                    ('Hashtag', hash_list),
                    ('URL', url_list)):

    # Build a frequency map for each set of data
    # and plot the values
    c = Counter(data)
    plt.hist(c.values())

    # Add a title and y-label ...
    plt.title(label)
    plt.ylabel("Number of items in bin")
    plt.xlabel("Bins (number of times an item appeared)")

    # ... and display as a new figure
    plt.figure()
```


Word
Number of items in bin
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
0
200
400
600
800
1000
1200
1400
1600
1800
Bins (number of times an item appeared)


Screen Name
Number of items in bin
0
200
400
600
800
1000
1200
0
50
100
150
200
250
Bins (number of times an item appeared)


Hashtag
Number of items in bin
0
200
400
600
800
1000
1200
1400
1600
1800
0
1000
2000
3000
4000
5000
6000
Bins (number of times an item appeared)


URL
Number of items in bin
0
500
1000
1500
2000
2500
3000
0
10
20
30
40
50
60
70
80
Bins (number of times an item appeared)

```
<matplotlib.figure.Figure at 0x11397a20>
```

```
In [5]:
```